A00274596
Carlos Arredondo

Para realizar el aprovisionamiento primero se crearon las carpetas y archivos como se encuentran el repositorio. Luego se modificó el archivo **Dockerfile** de la carpeta fluentd para seleccionar la imagen de docker con la que se trabajó.

```
# fluentd/Dockerfile
FROM fluent/fluentd:v0.12-debian
RUN echo "fluentd"
RUN ["gem", "install", "fluent-plugin-elasticsearch", "--no-rdoc", "--no-ri", "--version", "1.9.2"]
RUN rm /fluentd/etc/fluent.conf
COPY ./conf/fluent.conf /fluentd/etc
```

Luego so configuró el archivo **fluent.conf** especificando parámetros y puertos necesarios para que los servicios puedan correr sin ningún inconveniente.

```
<source>
  @type forward
  port 24224
  bind 0.0.0.0
</source>

<filter *.**>
  @type parser
  format apache
  key_name log
</filter>


<match tutum>
  @type copy
   <store>
    @type file
    path /fluentd/log/tutum.*.log
    time_slice_format %Y%m%d
    time_slice_wait 10m
    time_format %Y%m%dT%H%M%S%z
    compress gzip
    utc
    format json
  </store>
  <store>
    @type elasticsearch
    host elasticsearch
    port 9200
    logstash_format true
    logstash_prefix logstash
    logstash_dateformat %Y%m%d
    include_tag_key true
    tag_key @log_name
    flush_interval 1s
  </store>
 <store>
    @type stdout
```

```
  @type stdout
 </store>
/match>
match visualizer>
 @type copy
  <store>
   @type file
   path /fluentd/log/visualizer.*.log
   time_slice_format %Y%m%d
   time_slice_wait 10m
   time_format %Y%m%dT%H%M%S%z
   compress gzip
   utc
   format json
 </store>
 <store>
   @type elasticsearch
   host elasticsearch
   port 9200
   logstash_format true
   logstash_prefix logstash
```

```
<match visualizer>
  @type copy
   <store>
    @type file
    path /fluentd/log/visualizer.*.log
    time_slice_format %Y%m%d
    time_slice_wait 10m
    time_format %Y%m%dT%H%M%S%z
    compress gzip
    utc
    format json
  </store>
  <store>
    @type elasticsearch
    host elasticsearch
    port 9200
    logstash_format true
    logstash_prefix logstash
    logstash_dateformat %Y%m%d
    include_tag_key true
    type_name access_log
    tag_key @log_name
    flush_interval 1s
  </store>
  <store>
     @type stdout
  </store>
</match>
```

Finalmente se crea el archivo **docker-swarm.yml** donde se establecen los parámetros como los del servicio web, donde se limita la memoria RAM y el porcentaje de uso de la CPU

```
version: "3"

services:

  whoami:
    image: tutum/hello-world
    networks:
      - net
    ports:
      - "80:80"
    logging:
      driver: "fluentd" # Logging Driver
      options:
        tag: tutum    # TAG
        # fluentd-address: 127.0.0.1:24224
    deploy:
      resources:
        limits:
          cpus: '0.10'
          memory: 20M
        reservations:
          cpus: '0.05'
          memory: 10M

      restart_policy:
           condition: on-failure
           delay: 20s
           max_attempts: 3
           window: 120s
      mode: replicated
      replicas: 3

      placement:
        constraints: [node.role == worker]
      update_config:
        delay: 2s
```

Como no se tenía acceso a otra máquina física se decidió crear una virtual con vagrant y se automatizó la instalación de docker para poder unirse al cluster creado en el sistema operativo anfitrión, a continuación se muestra el código del **Vagrantfile**

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.ssh.insert_key = false
  config.vm.network "forwarded_port", guest: 80, host: 1234, host_ip: "127.0.0.1"
  config.vm.define :centos_prueba do |server|
    server.vm.box = "centos/7"
    server.vm.provider :virtualbox do |vb|
      vb.customize ["modifyvm", :id, "--memory", "1024","--cpus", "1", "--name", "centos_prueba" ]
    end
    server.vm.provision "shell", inline: <<-SHELL
       sudo yum -y remove docker
       sudo yum -y remove docker-selinux
       sudo rpm --import "https://sks-keyservers.net/pks/lookup?op=get&search=0xee6d536cf7dc86e2d7d56f59a178ac6c6238f52e"
       sudo yum install -y yum-utils
       sudo yum-config-manager --add-repo https://packages.docker.com/1.13/yum/repo/main/centos/7
       sudo yum makecache fast
       sudo yum -y install docker-engine
       sudo systemctl start docker
       sudo usermod -aG docker vagrant
    SHELL
  end
end
```

con el comando **docker swarm init –advertise-addr 192.168.1.34** se crea nodo manager en el sistema operativo anfitrión. Con el comando **docker build -t parcial2.1/myfluentd:latest .** Se construye la imagen de docker y desde el directorio donde se encuentra guardado el archivo docker-swarm.yml se corre el comando **docker stack deploy -c docker-swarm.yml test** para desplegar el servicio.

Una vez desplegado el servicio, en el directorio donde se encuentra el archivo **Vagrantfile** se corre el comando **vagrant up** para subir la maquina virtual con la instalación de docker automatizada, después con el comando **vagrant ssh** se accede a la máquina virtual y una vez en ella se corre el comando que se muestra a continuación para unirse al clúster creado en el sistema anfitrión.

```
[vagrant@localhost ~]$ sudo su
[root@localhost vagrant]# docker swarm join --token SWMTKN-1-2ghibgvbk7e83odefpl
67023ftxjy104edhg66ybw4e0czzkkn-97ockwsklk9tu598gei1isi9n 192.168.1.34:2377
This node joined a swarm as a worker.
[root@localhost vagrant]#
```

Finalmente no se pudo encontrar la razón por la cual los servicios de elasticsearch no corrieron en la maquina local como se puede evidenciar en la siguiente captura

Status: Red

6a378035df0f

Heap Total (MB)	Heap Used (MB)	Load
60.29	51.83	1.29, 1.42, 1.27

Response Time Avg (ms)	Response Time Max (ms)	Requests Per Second
1.07	10.42	0.47

Status Breakdown

ID	Status
ui settings	Elasticsearch plugin is red
plugin:kibana@5.6.9	Ready
plugin:elasticsearch@5.6.9	Unable to connect to Elasticsearch at http://elasticsearch:9200.
plugin:console@5.6.9	Ready
plugin:metrics@5.6.9	Ready
plugin:timelion@5.6.9	Ready